

**NATIONAL INSTITUTE OF TECHNOLOGY,**  
**ROURKELA**



**Submission of project report for the evaluation of the final year  
project**

Titled

**Particle Swarm Optimization**  
**Applied to**  
**Job Shop Scheduling**

Under the guidance of:  
Prof. S. S. Mahapatra  
Dept. of Mechanical Engineering  
NIT, Rourkela.

Submitted by:  
Vivek Vishal  
10303031  
B.Tech IV,  
Dept. of Mechanical Engg.  
NIT, Rourkela

Submitted by:  
Piyush Dwivedi  
10303055  
B.Tech IV,  
Dept. of Mechanical Engg.  
NIT, Rourkela

**NATIONAL INSTITUTE OF TECHNOLOGY,**  
**ROURKELA**



**Submission of project report for the evaluation of the final year  
project**

Titled

**Particle Swarm Optimization**  
**Applied to**  
**Job Shop Scheduling**

Under the guidance of:  
Prof. S. S. Mahapatra  
Dept. of Mechanical Engineering  
NIT, Rourkela.

Submitted by:  
Vivek Vishal  
10303031  
B.Tech IV,  
Dept. of Mechanical Engg.  
NIT, Rourkela

Submitted by:  
Piyush Dwivedi  
10303055  
B.Tech IV,  
Dept. of Mechanical Engg.  
NIT, Rourkela



**National Institute of Technology  
Rourkela**

**CERTIFICATE**

This is to certify that the thesis entitled, “PARTICLE SWARM OPTIMIZATION APPLIED TO JOB SHOP SCHEDULING” submitted by Sri PIYUSH DWIVEDI and Sri VIVEK VISHAL in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Mechanical Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

**Dr. S.S. Mahapatra**  
Dept. of Mechanical Engineering  
NIT, Rourkela.

## **ACKNOWLEDGEMENT**

I wish to express my deep sense of gratitude and indebtedness to Dr. S. S. Mahapatra, Department of Mechanical Engineering, N.I.T Rourkela for introducing the present topic and for his inspiring guidance, constructive criticism and valuable suggestion throughout this project work.

I would like to express my gratitude to Dr. B.K. Nanda (Head of the Department) for his constant support and encouragement. I am also thankful to all staff members of Department of Mechanical Engineering NIT Rourkela.

30<sup>TH</sup> April 2007

Vivek Vishal  
Roll: 10303031  
Piyush Dwivedi  
Roll: 10303055

## **Abstract**

In this project we have to apply the particle swarm optimization algorithm to job shop scheduling problem. Job shop scheduling is a combinatorial optimization problem where we have to arrange the jobs which may or may not be processed in every machine in a particular sequence and each machine has a different sequence of jobs. Job shop scheduling is a complex extended version of flow shop scheduling which is a problem where each job is processed through each and every machine and each machine has a same sequence of jobs. Our main objective in both kind of problem is to arrange the jobs in a sequence which gives minimum value of make span.

PSO (Particle swarm optimization) helps us to find a combination of job sequence which has the least make span. In PSO a swarm of particles which have definite position and velocity for each job. In PSO, to find the combinations we use a heuristic rule called Smallest Position Value (SPV). According to smallest position value rule jobs are arranged in ascending order of their positions i.e. job having least position value is put first in sequence.

In this project PSO is first applied to flow shop scheduling problem. This is done to understand how PSO algorithm can be applied to scheduling problem as flow shop scheduling problem is a simple problem. After Understanding the PSO algorithm, the algorithm is extended to apply in job shop scheduling problem for  $n$  jobs and  $m$  machines.

# CONTENTS

	Page No
<i>Abstract</i>	<i>iv</i>
<b>Chapter 1</b>	<b>GENERAL INTRODUCTION</b>
1.1	Introduction 1-2
1.2	Sequencing and Scheduling 2
1.3	Types of Scheduling 2-3
1.3.1	Single machine scheduling 3
1.3.2	Flow shop scheduling 3
1.3.3	Job shop scheduling 3
<b>Chapter 2</b>	<b>SEQUENCING AND SCHEDULING</b>
2.1	Some method for Scheduling and Sequencing 4-12
2.1.1	Johnson's Rule 4-5
2.1.2	NEH Algorithm 5-6
2.1.3	Particle swarm optimization 7-10
2.1.4	Ants colony optimization 10-12
2.2	Comparison between NEH, PSO and ACO 12-15
2.2.1	Inference of graphs 15
<b>Chapter 3</b>	<b>JOB SHOP SCHEDULING AND ITS REPRESENTATIONS</b>
3.1	Calculation of make span in job shop scheduling problem 16-17
3.2	Types of job shop scheduling representations 18-22
3.2.1	Operation based representation 18
3.2.2	Job based representation 19-20
3.2.3	Preference list based representation 20-21
3.2.4	Completion time based representation 22
3.2.5	Machine based representation 22
<b>Chapter 4</b>	<b>ALGORITHM AND RESULTS</b>
4.1	Basic terms of PSO Algorithm applied to Job shop scheduling 23
4.2	Algorithm for PSO applied to Job shop scheduling 24
4.3	Result of PSO algorithm applied to Job shop scheduling 25-27
4.3.1	For 3 X 3 problem 25
4.3.2	For 10 X 10 problem 26-27
<b>Chapter 5</b>	<b>CONCLUSION</b> 28-29
	<b>REFERENCES</b> 30

# Chapter 1

## **GENERAL INTRODUCTION**

## **1.1 INTRODUCTION:**

Machine scheduling problems arise in diverse areas such as flexible manufacturing system, production planning, computer design, logistics, communication etc. A common feature of many of these problems is that no efficient solution algorithm is known yet for solving it to optimality in polynomial time. The classical job shop scheduling problem is one of the most well known scheduling problems. Informally the problem can be described as follows:

There are set of jobs and a set of machines. Each job consists of chain of operation, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. Each machine can process at most one operation at a time. A schedule is an allocation of operations to time intervals of the machines. The problem is to find the schedule of minimum length.

JSP is among the hardest combinatorial optimization problems. Because of its inherent intractability, heuristic procedures are an attractive alternative. Most conventional heuristic procedures use a priority rule, which is a rule for choosing operation from specified subset of as yet unscheduled operations.

In this project we have to study the method of Particle swarm optimization (PSO) which is being applied to job shop scheduling case so as to get an optimum processing time.

Particle swarm optimization is the latest evolutionary optimization techniques, and it is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. PSO does not employ the filtering operations like crossover or mutation. In this search procedure, the members of the entire population are maintained so that the information is socially shared among individuals to direct the search towards the best position in search space.

In a PSO algorithm, each member is called particle, and each particles has some velocity with which it flies in the search space, which is being upgraded by the particle's own experience and the experience of the particle's neighbors or particle's experience from the whole group, here regarded as swarm. There are two types of the PSO algorithm, namely PSO with a global neighborhood and PSO with a local neighborhood.



According to the global neighborhood, each particle moves towards its best previous position and towards the best particle in the whole swarm called gbest model. On the other hand, according to the local variant so called lbest, each particle moves towards its previous position and towards the best particle in its restricted neighborhood. Simple concept and economic computational cost are the merits of PSO, which is a combinatorial optimization problem technique.

Job shop scheduling problem is a typical combinatorial optimization problem, in job shop scheduling problem each and every job is not processed through all machines in the same sequence as in flow shop scheduling problem. Here different jobs have different sequence of operations and jobs may or may not pass through every machine and each machine has different sequence of jobs. So it is a complex combinatorial problem in which different kind of representation can be done but we employ job based representation. There are several constraints on jobs and machines in job shop scheduling problems which are as follows:

1. A job does not visit the same machine twice.
2. There are no precedence constraints among operations of different jobs.
3. Operations can not be interrupted.
4. Each machine can process only one job at a time.
5. Neither release times nor due dates are specified.

### **1.2 Sequencing and Scheduling:**

Sequencing is a technique to order the jobs in a particular sequence. There are different types of sequencing which are followed in industries such as first in first out basis, priority basis, job size basis and processing time basis etc. In processing time basis sequencing for different sequence, we will achieve different processing time. The sequence is adapted which gives minimum processing time.

By Scheduling, we assign a particular time for completing a particular job. The main objective of scheduling is to arrive at a position where we will get minimum processing time.

### **1.3 Types of Scheduling:**

Basically there are three types of scheduling:

### **1.3.1 Single Machine Schedule:**

Here we arrange the order of jobs in a particular machine. We achieve the best result when the jobs are arranged in the ascending order of their processing time i.e. the job having least processing time is put first in sequence and processed through the machine and the job having maximum processing time is put last in sequence.

### **1.3.2 Flow Shop Scheduling:**

It is a typical combinatorial optimization problem, where each job has to go through the processing in each and every machine on the shop floor. Each machine has same sequence of jobs. The jobs have different processing time for different machines. So in this case we arrange the jobs in a particular order and get many combinations and we choose that combination where we get the minimum make span.

### **1.3.3 Job Shop Scheduling:**

It is also a typical combinatorial optimization problem, but the difference is that, here all the jobs may or may not get processed in all the machines in the shop floor i.e. a job may be processed in only one or two machines or a different job may have to go through the processing in all the machine in order to get completed. Each machine has different sequence of jobs. So it is a complex web structure and here also we choose that combination of arrangements that will be giving the least make span.

# Chapter 2

## **SEQUENCING AND SCHEDULING**

## **2.1 Some methods for Scheduling and Sequencing:**

### **2.1.1 Johnson's Rule:**

This rule is one of the simplest one to find out the minimum total completion time. It is a thumb rule and is basically employed to two machines. In case of three machines, this rule is applied only when either of the following conditions is satisfied.

- a. Minimum machining time of machine 1 should be greater than or equal to maximum machining time of machine 2.
- b. Minimum machining time of machine 3 should be greater than or equal to maximum machining time of machine 2.

### **Algorithm for Johnson's Rule:**

- i. First of all different processing times for different jobs are assigned for both machines.
- ii. The least processing time of a job from all the jobs is detected and if the minimum processing time of a particular job is on machine 1, then the job is put in the first sequence otherwise if the minimum processing time is on machine 2, then it is put in last place of the sequence.
- iii. The selected job is then discarded from the sequence.
- iv. Then again the minimum processing time of any job from the list of jobs is detected and if the least processing time is on machine 1, and then it is put accordingly in the second position if the previous job is in first position or in the first position if the previous job is in last position. While if the least processing time is on machine 2, then it is put accordingly in the last position if the previous job is in the first position otherwise in the second last position.
- v. Accordingly all the jobs are sequenced in the above mentioned procedure.

For Example:

Consider 5 jobs and 2 machine scheduling problem with given processing times as follows:

Table-2.1

Jobs	Processing time in m/c 1	Processing time in m/c 2
1	8	5
2	3	2
3	1	4
4	5	3
5	2	1

From the above mentioned algorithm sequence is evaluated which is as follows:

Sequence:

Table-2.2

3	1	4	2	5
---	---	---	---	---

### **2.1.2 NEH heuristic method:**

Nawaz, Ensore and Ham [29] develop a constructive heuristic method for the permutation flow shop make span problem, called NEH algorithm. It is based on the idea that a job with a high total operating time on the machines should be placed first at an appropriate relative order in the sequence. Thus, jobs are sorted in non-increasing order of their total operating time requirements. The final sequence is built in a constructive way, adding a new job at each step and finding the best partial solution. For example, the NEH algorithm inserts a third job into the previous partial solution that gives the best objective function value under consideration. However, the relative position of the two previous job sequence remains fixed. The algorithm repeats the process for the remaining jobs according to the initial ordering of the total operating time requirements. Again, to apply the NEH algorithm to the flexible flow shop problem with unrelated parallel machines, the total operating times for calculating the job sequence for the first stage are calculated for the nine combinations of relative speeds of machines and setup times. Contrary to the algorithms presented before, the NEH algorithm constructs job sequences by considering the minimization of the convex combination of make span and tardy number of jobs.

The NEH algorithm is usually applied to provide the initial solution for an improvement method such as a tabu search (Nowicki and Smutnicki, 1996; Grabowski and Pempera, 2001; Grabowski and Wodecki, 2004) or a genetic algorithm (Wang and Zheng, 2003). The main drawback of the NEH algorithm is that a total of  $[n(n + 1)/2] - 1$  partial

schedules need to be evaluated. Therefore, the running time of the NEH algorithm increases rapidly as the problem size increases. If the running time could be reduced, then the efficiency of those improvement methods that take initial solutions from NEH could also be improved.

The pure heuristic algorithm is NEH (Nawaz, Ensore, & Ham 1983), which is widely regarded as the good performing heuristic for the PFSP (Taillard 1990). Despite its simplicity NEH produces reasonably good solutions to Taillard's benchmark problems. However, the solutions produced by path re-link

### **Algorithm for NEH applied to flow shop scheduling:**

- i. The total processing time for each job is calculated based on their sequence and stored in an array.
- ii. The jobs are sequenced according to the descending order of their sums of total job processing times on the machines. It means job having highest processing time for all the operations is put first on the list and job having least processing time for all the operations is placed last on the list.
- iii. The first two jobs from the above list (i.e. the jobs having highest processing time and 2<sup>nd</sup> highest processing time) are considered and scheduled.
- iv. Scheduling is done in two ways either first job is placed first in the order of scheduling or second job is placed first in order of scheduling, The sequence which gives minimum partial make span as if there were only two jobs is adopted.
- v. The 3rd job from the list is inserted into the location in the partial schedule.
- vi. Now scheduling is done considering all the possibilities but not changing the sequence which is been found in step 4. As for example if let the previous sequence [2, 1] gives least partial make span then possibilities which are to be considered are [2,1,3], [2,3,1], [3,2,1].
- vii. Among all the possible ways which gives least value of partial make span is adopted.
- viii. The above 3 steps are repeated until all the jobs are not scheduled.
- ix. The least value of make span is calculated and the sequence of jobs for which the value of make span is minimum is adopted

### **2.1.3 Particle Swarm Optimization:**

In PSO the evaluation of each particle in the swarm requires the determination of the permutation of jobs for the flow shop and here we use a heuristic rule called **Smallest Position Value (SPV)** to enable the continuous PSO algorithm to be applied to all classes of sequencing problem.

The basic terms used in PSO algorithm are as follows:

- Particle:  $X_i$  denotes the  $i$ th particle in the swarm at a particular iteration and is represented by  $n$  number of dimensions as  $X_i = [x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}]$ , where  $x_{ij}$  is the position value of  $i$ th particle with respect to  $j^{\text{th}}$  dimension.
- Population:  $\text{Pop}_k$  is the set of  $r$  particles in the swarm at  $k^{\text{th}}$  iteration i.e.  $\text{Pop}_k = [X_1, X_2, X_3, \dots, X_r]$ .
- Permutation:  $\text{Pm}_i$  is the permutation of jobs implied by particle  $X_i$ .  $\text{Pm}_i = [\text{pm}_{i1}, \text{pm}_{i2}, \text{pm}_{i3}, \dots, \text{pm}_{in}]$  where  $\text{pm}_{ij}$  is the assignment of job  $j$  of the particle  $i$  in the permutation  $\text{Pm}_i$  at a particular iteration with respect to  $j^{\text{th}}$  dimension.
- Particle Velocity:  $V_i$  denotes the velocity of  $i$ th particle at a particular iteration. It can be identified  $V_i = [v_{i1}, v_{i2}, v_{i3}, \dots, v_{in}]$ , where  $v_{ij}$  is the velocity of  $i^{\text{th}}$  particle with respect to  $j^{\text{th}}$  dimension.
- Inertia Weight:  $W^k$  is a parameter to control the impact of the previous velocity on the current velocity.
- Personal Best:  $P_i$  represents the best position of the particle  $i$  with the best fitness until iteration  $k$ . So, the best position associated with the best permutation and fitness value of the particle  $i$  obtained so far is called the personal best. For each particle,  $P_i$  can be determined and updated at each iteration.
- Global Best:  $G^k$  denotes the best position of the globally best particle achieved so far in the whole swarm.

### **Algorithm for PSO applied to flow shop scheduling:**

- i. Initialization: Set  $k=0$ ,  $r$ = twice the no. of jobs. The position and velocity of each particle is generated randomly from the given dimensions and this is done with respect to each job.
- ii. SPV rule is applied: For each particle the permutations of jobs are found out by SPV rule which is the method of arranging the jobs in the ascending order of their particle's position.
- iii. Make span Evaluation: The make span for each permutation is calculated and the particle having the least make span becomes the personal best for that iteration.
- iv. The counter is upgraded to next iteration.  $k=k+1$ .
- v. The inertia weight is upgrade by the formula  $w^k=w^{k-1}\alpha$ , ( $\alpha$ - decrement factor).
- vi. The velocity is updated  $V_{ij}^k = w^{k-1} V_{ij}^{k-1} + c_1 r_1 (p_{ij}^{k-1} - x_{ij}^{k-1}) + c_2 r_2 (g_{ij}^{k-1} - x_{ij}^{k-1})$ .  
Where  $c_1$ ,  $c_2$  are social and cognitive parameters,  $r_1$ ,  $r_2$  are random numbers between (0,1).
- vii. The position is updated  $x_{ij}^k = x_{ij}^{k-1} + V_{ij}^k$ .
- viii. For the updated particle position, we have found out the permutation of jobs for each particle by the SPV rule.
- ix. The new personal best is found out and is compared with previous personal best and if it comes or has low value then it is updated as the personal best.
- x. The minimum value of personal best among all the personal best gives the global best, and that arrangement of jobs which gives the global best will be adopted.
- xi. If the no. of iteration exceeds the maximum number of iteration, or maximum CPU time, then we should stop.



## **Sample Result of PSO applied to job shop scheduling for 10x10 problems:**

Data.in <Input File>

Table-2.3

1.045	0.393	33.180	3.293	35.217	21.503	53.168	19.382	69.334	94.892
27.204	43.985	10.789	69.125	55.870	4.109	16.351	80.733	67.868	75.675
81.396	94.994	21.723	42.253	94.308	83.114	91.401	80.275	47.652	59.871
65.506	59.372	54.369	71.252	11.282	40.214	12.022	66.454	47.015	48.619
55.861	34.105	34.105	26.177	17.802	41.885	68.784	60.206	53.293	63.904
61.717	0.302	0.302	26.624	45.646	38.268	37.304	57.532	59.744	27.703
16.886	79.712	79.712	33.056	40.456	23.802	40.984	5.375	48.915	97.387
0.160	40.573	40.573	7.318	25.177	96.492	35.156	40.087	19.518	19.512
20.956	24.654	24.654	88.048	89.634	72.856	45.670	52.626	3.493	12.928
45.344	47.915	47.915	25.068	11.345	63.433	64.279	87.739	86.682	4.233

Data.out <Output File>

For iter 1 gbest=1052.635

For iter 2 gbest=1052.635

For iter 3 gbest=1052.635

For iter 4 gbest=1052.635

For iter 5 gbest=1052.635

For iter 6 gbest=1052.635

For iter 7 gbest=1052.635

For iter 8 gbest=1052.635

For iter 9 gbest=1052.635

For iter 10 gbest=1052.635

For iter 11 gbest=1052.635

For iter 12 gbest=1052.635

For iter 13 gbest=1052.635  
For iter 14 gbest=1050.831  
For iter 15 gbest=1050.831  
For iter 16 gbest=1050.831  
For iter 17 gbest=1050.831  
For iter 18 gbest=1050.831  
For iter 19 gbest=1050.831  
For iter 20 gbest=1050.831  
For iter 21 gbest=1050.831  
For iter 22 gbest=1047.071  
For iter 23 gbest=1047.071  
For iter 24 gbest=1047.071  
For iter 25 gbest=1047.071  
For iter 26 gbest=1047.071  
For iter 27 gbest=1047.071  
For iter 28 gbest=1047.071  
For iter 29 gbest=1047.071  
For iter 30 gbest=1047.071

#### **2.1.4 Ants Colony Optimization:**

The main idea in ant-colony algorithms is to mimic the pheromone trail used by real ants as a medium for communication and feedback among ants. Basically, the ACO algorithm is a population-based, cooperative search procedure that is derived from the behavior of real ants. ACO algorithms make use of simple agents called ants that iteratively construct solutions to combinatorial optimization problems. The solution generation or construction by ants is guided by (artificial) pheromone trails and problem-specific heuristic information. ACO algorithms can be applied to combinatorial optimization problems by designing solution components, which the ants use to iteratively construct solutions, and in the process, the ants deposit pheromone. Basically, in the context of combinatorial optimization problems, pheromones indicate the intensity of ant-trails with respect to

solution components, and such intensities are determined on the basis of the influence or contribution of each solution component with respect to the objective function. An individual ant constructs a complete solution by starting with a null solution and iteratively adding solution components until a complete solution is constructed. After the construction of a complete solution, every ant gives feedback on the solution by depositing pheromone (i.e., updating trail intensity) on each solution component. Typically, solution components which are part of better solutions or used by ants over many iterations will receive a higher amount of pheromone, and hence, such solution components are more likely to be used by the ants in future iterations of the ACO algorithm. This is achieved by additionally making use of pheromone evaporation in updating trail intensities (Stuetzle and Hoos, 2000).

It is to be noted that one can make use of only one ant in every iteration of the ACO algorithm, instead of using many ants in parallel in one iteration. In the former case, the number of iterations in the algorithm may increase, while the latter case may result in increased computational complexity. The consideration of parallel ants in one iteration is highly relevant in the context of parallel processor computing systems because the updatation of pheromone trails due to one ant can be rejected on the intensity of trails of other ants working in parallel. On the other hand, the consideration of one single ant renders the task of implementing the ACO algorithm on a single-process computing system easy and less complex. In addition, a complete solution that has been constructed by a single ant can be subjected to an improvement or local search scheme to unearth possibly the best solution in the neighborhood. Such a consideration of a single ant in ACO algorithms is quite common (e.g. Stuetzle, 1998).

### **Algorithm for Ant Colony Optimization:**

Initialization of the make span (from NEH algorithm);

Initialization of pheromone trail intensity;

While (termination condition not met)

{

    Calculate element probabilities of placing a job in a position;

    Choose the next job based on the probability;

Generate the initial sequence;  
 Calculate the make span;  
 Apply job-index- based local search;  
 Update the sequence & the make span based upon the local search;  
 Update global pheromone;  
 }

## **2.2 Comparison between PSO, NEH and ACO algorithm:**

The following comparison between PSO, NEH, ACO algorithm for flow shop scheduling problem is on the basis of make span. First we fix no. of jobs as constant and vary the value of no. of machines and then No. of machines is fixed and values of no. of jobs is varied and the following results have been evaluated.

Table 2.4

No. of jobs	No. of machines	ANTS	PSO	NEH
5	6	580	584.7	584.7
	10	842	849.3	858.1
	15	1089	1098	1203
	20	1300	1310	1482
	25	1563	1576	1740
	30	1905	1920	2141
	35	2160	2174	2473
	40	2567	2591	2768
10	6	591	604.7	604.7
	10	1089	1050	1232
	15	1457	1421	1529
	20	1726	1681	1925
	25	1912	1895	2105
	30	2207	2113	2298
	35	2585	2507	2712
	40	2923	2863	3346

Table 2.5

No. of machines	No. of jobs	ANTS	PSO	NEH
10	5	842	849.3	858.1
	10	1000	1017	1019
	15	1249	1374	1685
	20	1659	1687	1999
	25	1864	1968	2228
	30	2055	2275	2553
	35	2374	2542	2923
	40	2776	2885	3231
20	5	1344	1352	1479
	10	1681	1726	1925
	15	1809	1844	1879
	20	2098	2124	2185
	25	2232	2303	2412
	30	2475	2562	2682
	35	2698	2735	2904
	40	2881	3004	3142

From the above two tables the results of all algorithms can be collected and compared from the following tables and graphs.

Table-2.6

No. of jobs	No. of m/c	PSO Avg. make span	ACO Avg. make span	NEH Avg. make span	PSO V/s NEH	ACO V/s NEH	PSO V/s ACO
5	10	798.955	792.3	803.349	0.55%	1.38%	0.83%
5	20	1352.264	1343.7	1478.68	8.55%	9.13%	0.63%
10	10	1088.965	1049.5	1232.488	11.64%	14.85%	3.62%
10	20	1725.782	1680.8	1924.622	10.33%	12.67%	2.61%

The above table gives us the comparative study between ACO, PSO and NEH algorithms.

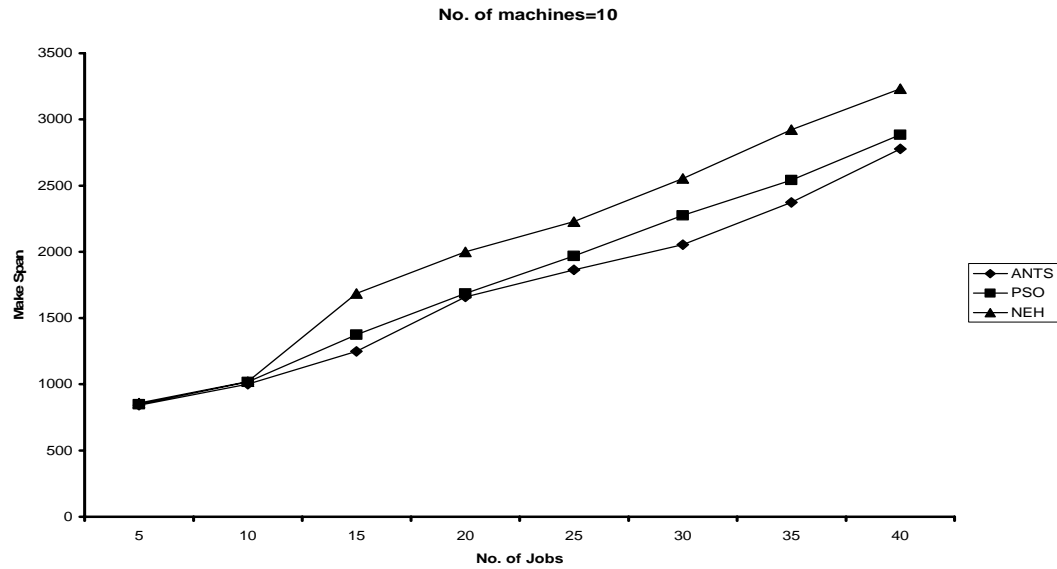


Fig-2.1

The above graph is the variation of make span with respect to no. of jobs keeping no. of machines fixed.

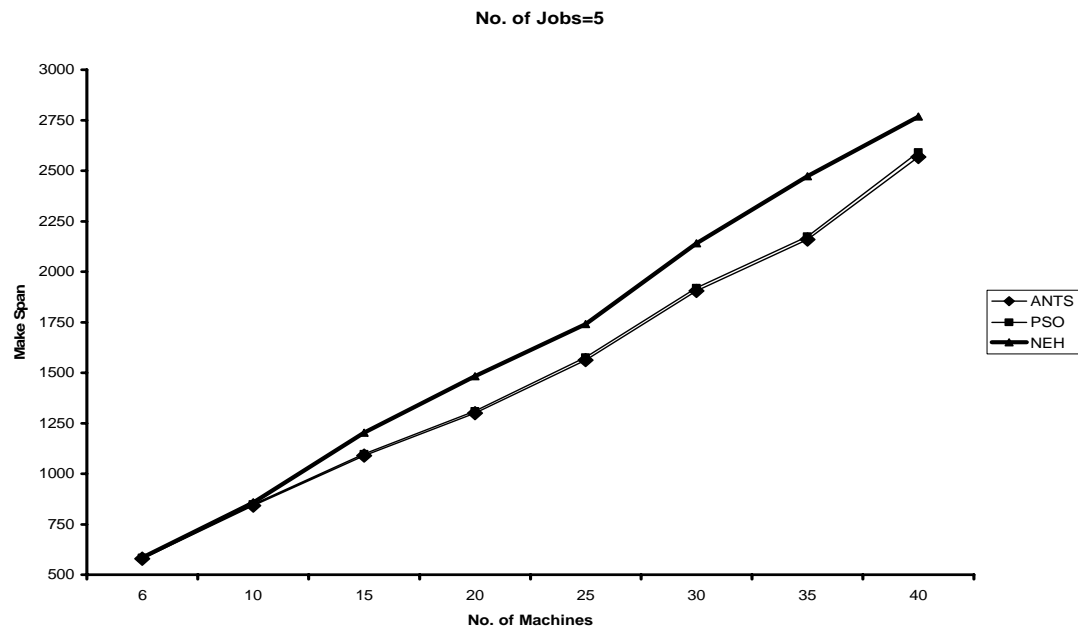


Fig 2.2

The above graph shows the variation of make span with respect to no. of machines keeping no. of jobs fixed.

### **2.2.1 Inference of the Graphs:**

- When the no. of jobs is fixed and the no. of machines is varied, the variation of make span between PSO (Particle Swarm Optimization) and ACO (Ant Colony Optimization) is very small i.e. incomparable whereas NEH shows a significant variation from the above two.
- When the no. of machines is fixed and no. of jobs is varied, then there can be noticeable variation between the make spans of ACO and PSO algorithm.
- When the no. of machines is very low, there is very less variation in make spans calculated from NEH, PSO and ACO algorithms.
- When the no. of jobs is very low and no. of machines is fixed then there is incomparable variation between the results of NEH, PSO and ACO algorithm.
- When the no. of jobs is increased, we can observe noticeable variation among all the three types of optimization technique.
- For same no. of iteration ACO gives better result or less make span than PSO algorithm, which further gives better result or less make span than NEH algorithm.

# Chapter 3

## **JOB SHOP SCHEDULING AND ITS REPRESENTATIONS**



### **3.1 Calculation of Make Span in Job Shop Scheduling Problem:**

Calculation of make span in job shop scheduling problem is a vital problem in employing PSO algorithm to job shop scheduling problem. Following example illustrates how make span can be calculated in job shop scheduling problem.

Consider a problem with 3 jobs and 3 machines, each job is processed through different sequence of machines and processing time for each job in each machine is different. Let the processing time and machine sequence of all three jobs be given below

Table-3.1

Jobs	Machines		
	Operation 1	Operation 2	Operation 3
1	1	2	3
2	1	3	2
3	2	1	3

Table-3.2

Jobs	Processing Time		
	Operation 1	Operation 2	Operation 3
1	3	3	3
2	2	3	4
3	3	2	1

From the above two tables Make span of the problem is calculated

Table-3.3

Jobs	Machine 1		Machine 2		Machine 3	
	In	Out	In	Out	In	Out
1	0	3	3	6	8	11
2	3	5	8	12	5	8
3	5	7	0	3	11	12

Make Span = 12 units.

Gantt chart of the above scheduling is given below. Gantt chart is graphical representation of scheduling of jobs, In Gantt Chart time is placed at the abscissa and machine number is placed at the ordinate.



Fig-3.1

From the above Gantt chart, the sequence of jobs in each machine can be found out. The sequence of jobs in different machine is represented as follows.

Table-3.4

Machines	Jobs		
1	1	2	3
2	3	1	2
3	2	1	3

### 3.2 TYPES OF JOB SHOP SCHEDULING REPRESENTATION:

#### 3.2.1 OPERATION BASED REPRESENTATION:

This representation encodes a schedule as a sequence of operations and each gene stands for one operation.

A schedule is decoded from a chromosome (pool of gene) with the following decoding procedure:

1. Firstly the pool of genes is translated to a list of ordered operations.
2. Then the list is schedule is generated by a one-pass heuristic based on the list. The first operation in the list is scheduled first, then the second operation and so on.

For example.

Table-3.5

Job	Operations		
	1	2	3
	Processing Time		
J <sub>1</sub>	3	2	2
J <sub>2</sub>	1	5	3
J <sub>3</sub>	3	2	3
	Machine sequence		
J <sub>1</sub>	M <sub>1</sub>	M <sub>2</sub>	M <sub>3</sub>
J <sub>2</sub>	M <sub>1</sub>	M <sub>3</sub>	M <sub>2</sub>
J <sub>3</sub>	M <sub>2</sub>	M <sub>1</sub>	M <sub>3</sub>

The above pool of genes can be translated into a unique list of ordered operation as given:-

{O<sub>211</sub>, O<sub>111</sub>, O<sub>122</sub>, O<sub>133</sub>, O<sub>223</sub>, O<sub>312</sub>, O<sub>321</sub>, O<sub>333</sub>}

Where O<sub>jim</sub> denotes the i<sup>th</sup> operation of job J on machine M.

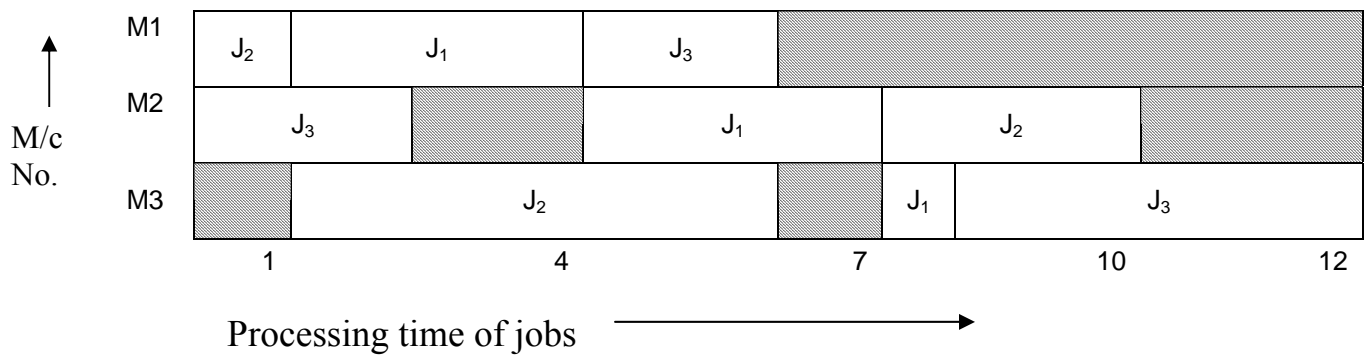


Fig-3.2

### 3.2.2 JOB-BASED REPRESENTATION:

This representation consists of a list of  $n$  jobs and a schedule is constructed according to the sequence of jobs. For a given sequence of jobs, all operations of the first job in the list are scheduled first, and then we consider the operation of the second job on the list. The first operation of the job under treatment is allocated in the best available processing time for all the corresponding machine the operations requires, and then the second operation and so on until all the operation are scheduled. The process is repeated with each of the job in the list considered in the appropriate sequence.

The Gantt chart of the same example as discussed previously by job based representation is as follows:

Let the order of the jobs is given as  $\{2\ 3\ 1\}$ . Then

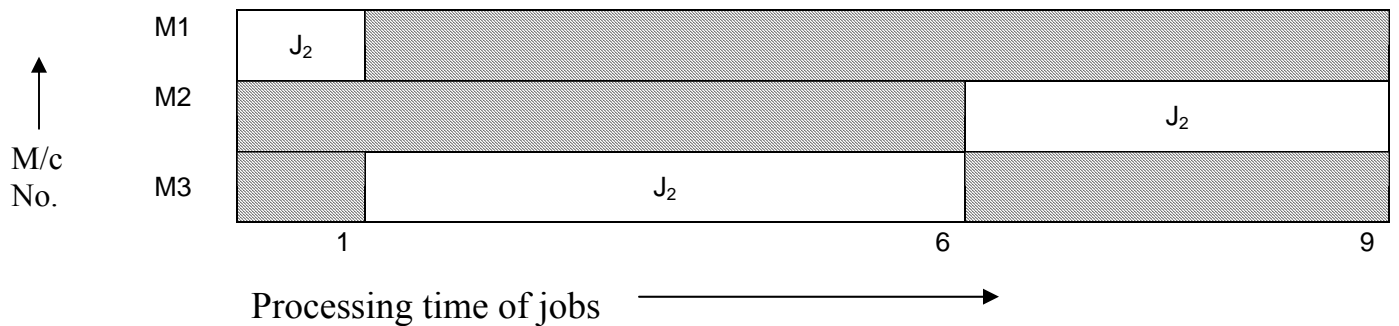


Fig-3.3  
Scheduling of first job  $J_2$



Fig-3.4  
Scheduling of second job  $J_3$

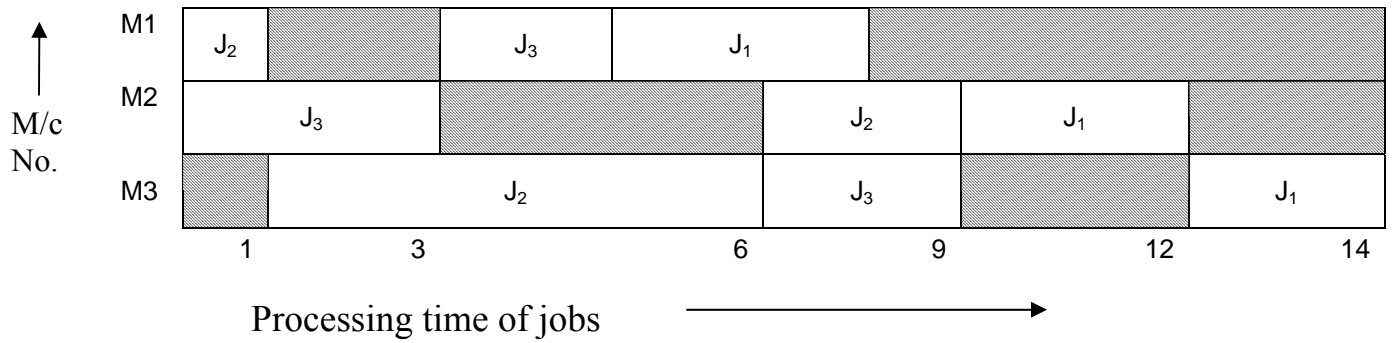


Fig-3.5  
Scheduling of third job J<sub>1</sub>

### 3.2.3 PREFERENCE LIST BASED OPERATION:

For each machine there is a different preference list and job scheduling is done according to that preference list only.

Considering the same previous example, the preference list on particular machine is given as:

{(2 3 1) (1 3 2) (2 1 3)} for machine m<sub>1</sub>, m<sub>2</sub> and m<sub>3</sub> respectively. This preference list shows that the first preferential operation are job j<sub>2</sub> on machine m<sub>1</sub>, job j<sub>1</sub> on machine m<sub>2</sub> and job j<sub>2</sub> on machine m<sub>3</sub>. So the Gantt chart is drawn below:

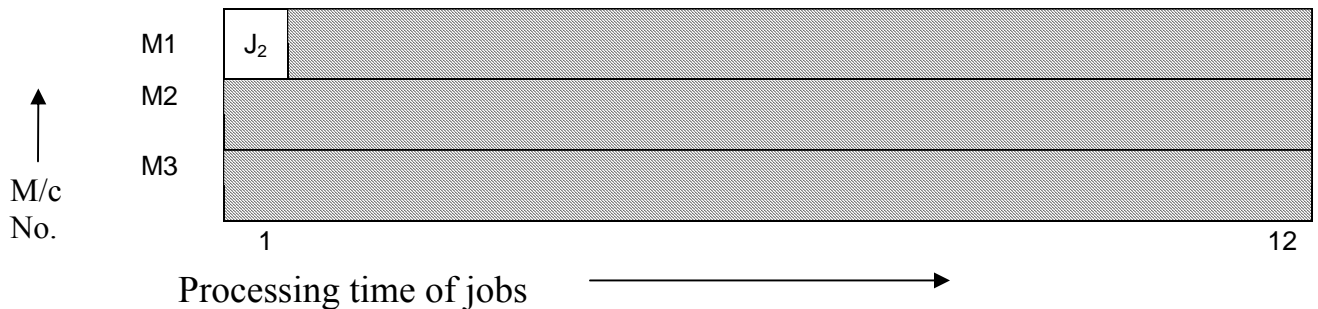


Fig-3.6  
Scheduling job J<sub>2</sub> on M<sub>1</sub>

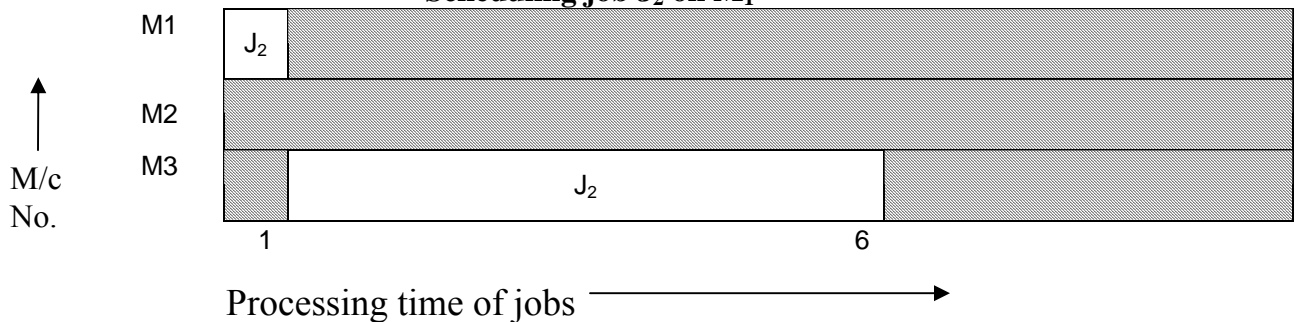


Fig-3.7  
Scheduling job J<sub>2</sub> on M<sub>3</sub>

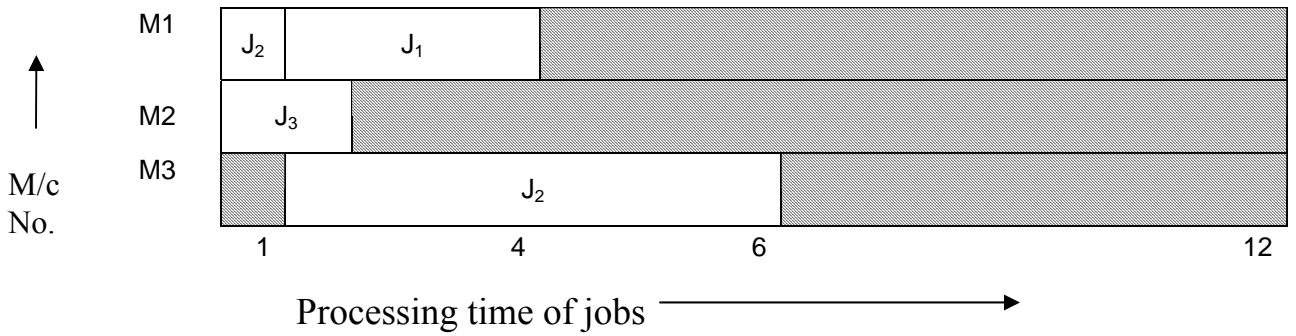


Fig-3.8  
Scheduling J<sub>1</sub> on M<sub>1</sub> and J<sub>3</sub> on M<sub>2</sub>

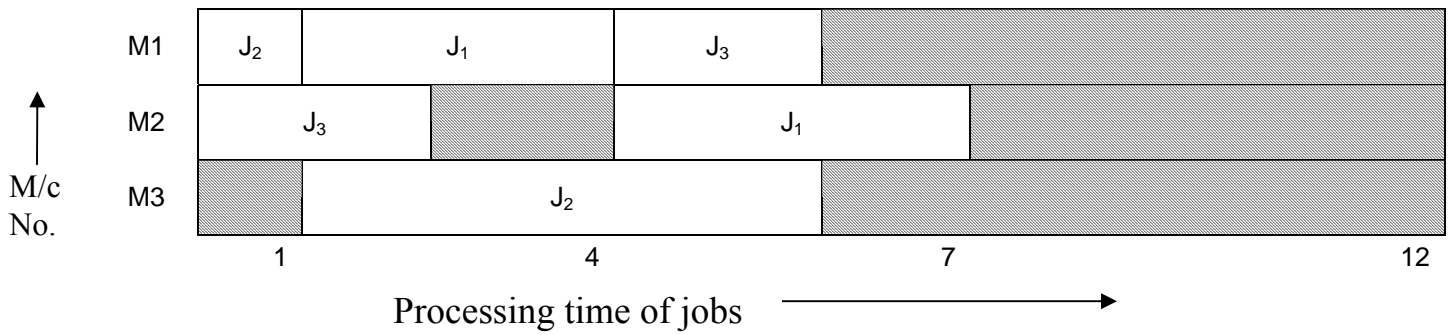


Fig-3.9  
Scheduling J<sub>3</sub> on M<sub>1</sub> and J<sub>1</sub> on M<sub>2</sub>

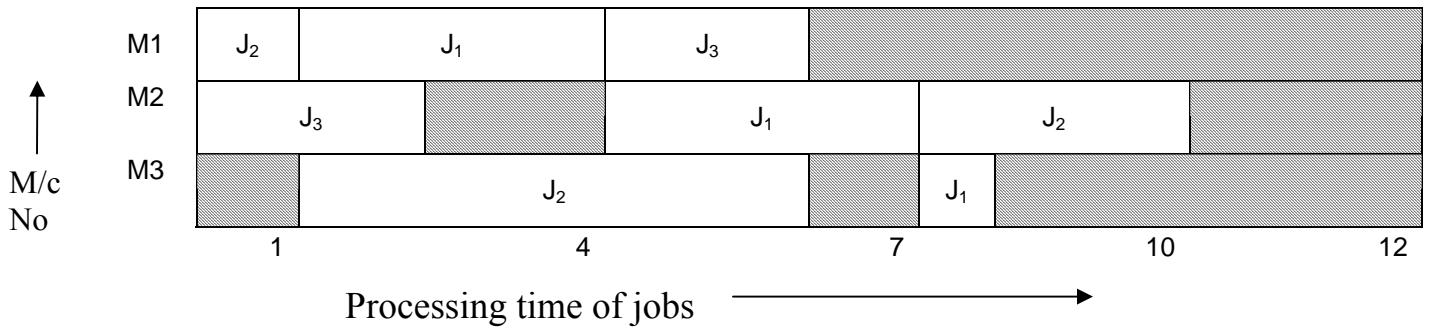


Fig-3.10  
Scheduling J<sub>3</sub> on M<sub>1</sub> and J<sub>1</sub> on M<sub>2</sub>

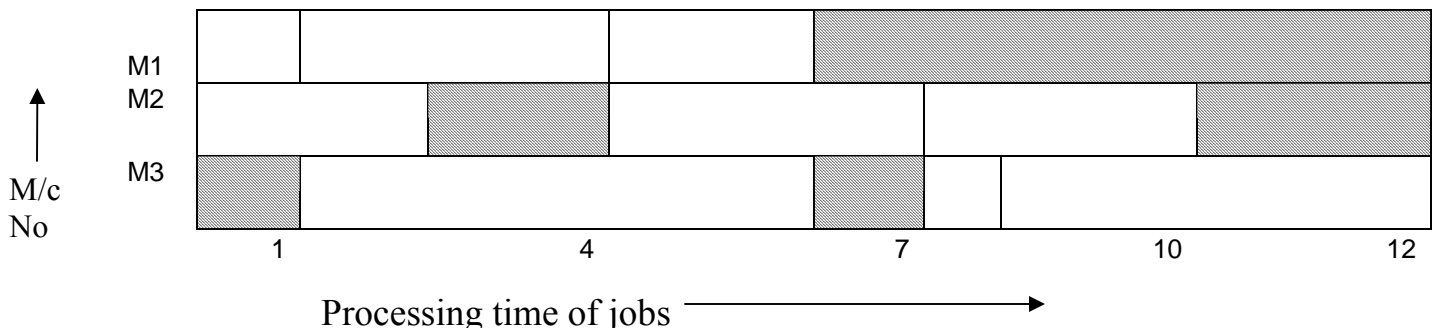


Fig-3.11 (Final Schedule)

### COMPLETION TIME BASED REPRESENTATION:

A pool of gene in this type of representation is an ordered list of completion times of operations. For the same example, the pool of genes can be represented as follows:

[ $C_{111}$   $C_{122}$   $C_{133}$   $C_{211}$   $C_{223}$   $C_{232}$   $C_{312}$   $C_{321}$   $C_{333}$ ]

Where  $C_{jim}$  means the completion time for operation  $i$  of job  $j$  on machine  $m$ . This representation is not suitable for most genetic Algorithms because it will yield an illegal schedule.

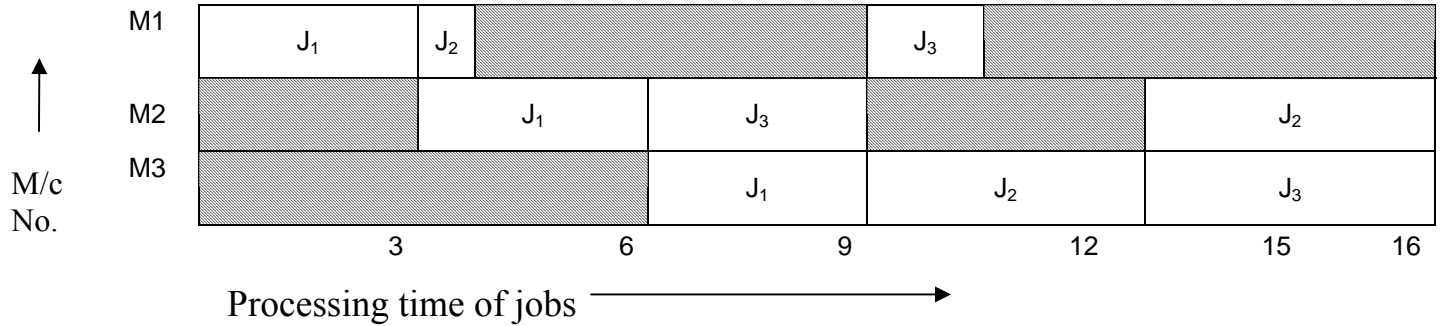


Fig-3.12

### 3.2.4 MACHINE BASED REPRESENTATION:

Here a pool of genes is encoded as a sequence of machines and a schedule is constructed with a shifting bottleneck heuristic based on the sequence. The shifting bottleneck heuristic is based on the idea of giving priority to bottleneck machines. Different measures of bottleneck quality of machines will yield a different sequence of bottleneck machines. It sequences the machine one by one successively, taking each time the machine that is identified as bottleneck among the machines not yet sequenced. Every time after a new machine is sequenced, all previously established sequence are reoptimised.

# Chapter 4

## **ALGORITHM AND RESULTS**



#### **4.1 Basic terms of PSO algorithm applied to Job shop scheduling problem:**

PSO algorithm applied to job shop scheduling problem is an extended version of PSO algorithm to flow shop scheduling problem. The basic terms which are used in the job shop scheduling are as follows

- Particle:  $X_i$  denotes the  $i$ th particle in the swarm at a particular iteration and is represented by  $n$  number of dimensions as  $X_i = [x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}]$ , where  $x_{ij}$  is the position value of  $i$ th particle with respect to  $j^{\text{th}}$  dimension.
- Population:  $\text{Pop}_k$  is the set of  $r$  particles in the swarm at  $k^{\text{th}}$  iteration i.e.  $\text{Pop}_k = [X_1, X_2, X_3, \dots, X_r]$ .
- Permutation:  $\text{Pm}_i$  is the permutation of jobs implied by particle  $X_i$ .  $\text{Pm}_i = [\text{pm}_{i1}, \text{pm}_{i2}, \text{pm}_{i3}, \dots, \text{pm}_{in}]$  where  $\text{pm}_{ij}$  is the assignment of job  $j$  of the particle  $i$  in the permutation  $\text{Pm}_i$  at a particular iteration with respect to  $j^{\text{th}}$  dimension.
- Particle Velocity:  $V_i$  denotes the velocity of  $i$ th particle at a particular iteration. It can be identified  $V_i = [v_{i1}, v_{i2}, v_{i3}, \dots, v_{in}]$ , where  $v_{ij}$  is the velocity of  $i^{\text{th}}$  particle with respect to  $j^{\text{th}}$  dimension.
- Inertia Weight:  $W^k$  is a parameter to control the impact of the previous velocity on the current velocity.
- Personal Best:  $P_i$  represents the best position of the particle  $i$  with the best fitness until iteration  $k$ . So, the best position associated with the best permutation and fitness value of the particle  $i$  obtained so far is called the personal best. For each particle,  $P_i$  can be determined and updated at each iteration.
- Global Best:  $G^k$  denotes the best position of the globally best particle achieved so far in the whole swarm.
- Machine Sequence:  $M_i$  denotes the machine sequence of  $i^{\text{th}}$  particle at a particular iteration. It can be represented as  $M_i = [m_{i1}, m_{i2}, m_{i3}, \dots, m_{in}]$  where  $m_{i1}$  is the machine sequence of the particle  $i$  with respect to  $j^{\text{th}}$  dimension.

#### **4.2 Algorithm for PSO applied to flow shop scheduling:**

- i. Initialization: Set  $k=0$ ,  $r$ = twice the no. of jobs. The position and velocity of each particle is generated randomly from the given dimensions and this is done with respect to each job.
- ii. SPV rule is applied: For each particle the permutations of jobs are found out by SPV rule which is the method of arranging the jobs in the ascending order of their particle's position.
- iii. Machine Sequence: For each job different machine sequences are taken as an input from input file. The jobs are scheduled according to this machine sequence.
- iv. Make span Evaluation: The make span for each permutation is calculated by using job based representation and the particle having the least make span becomes the personal best for that iteration.
- v. Job based representation: According to this representation a schedule is constructed according to the sequence of jobs. For a generated sequence of jobs all the operations of 1<sup>st</sup> job in the list are scheduled first, and then the operations of the second job in the list is considered and so on.
- vi. The counter is upgraded to next iteration.  $k=k+1$ .
- vii. The inertia weight is upgrade by the formula  $w^k = w^{k-1} \times \alpha$ , ( $\alpha$ - decrement factor).
- viii. The velocity is updated  $V_{ij}^k = w^{k-1} \times V_{ij}^{k-1} + c_1 r_1 (p_{ij}^{k-1} - x_{ij}^{k-1}) + c_2 r_2 (g_{ij}^{k-1} - x_{ij}^{k-1})$ .  
Where  $c_1$ ,  $c_2$  are social and cognitive parameters,  $r_1$ ,  $r_2$  are random numbers between (0,1).
- ix. The position is updated  $x_{ij}^k = x_{ij}^{k-1} + V_{ij}^k$ .
- x. For the updated particle position, we have found out the permutation of jobs for each particle by the SPV rule. Here the sequence of jobs is changed.
- xi. The new personal best is found out and is compared with previous personal best and if it comes or has low value then it is updated as the personal best.
- xii. The minimum value of personal best among all the personal best gives the global best, and that arrangement of jobs which gives the global best will be adopted.
- xiii. If the no. of iteration exceeds the maximum number of iteration, or maximum CPU time, then we should stop.

### **4.3 Results of PSO algorithm applied to job shop scheduling**

#### **4.3.1 For 3x3 problem:**

Mac\_Seq1.in <Input File for Machine Sequence>

Table 4.1

1	2	3
1	3	2
2	1	3

PSO1.in <Input File for Processing Time>

Table 4.2

15	91	64
11	55	42
69	76	26

PSO1.out <Output File>

For iter 1 gbest=370

For iter 2 gbest=370

For iter 3 gbest=370

For iter 4 gbest=370

For iter 5 gbest=370

For iter 6 gbest=370

For iter 7 gbest=362

For iter 8 gbest=362

For iter 9 gbest=362

For iter 10 gbest=362

For iter 11 gbest=362

For iter 12 gbest=362

For iter 13 gbest=362

For iter 14 gbest=362

For iter 15 gbest=362

### 4.3.2 For 10x10 problem:

Mac\_Seq2.in <Input File for Machine Sequence>

Table 4.3

9	6	1	5	4	7	3	2	0	0
5	6	4	7	3	9	8	0	0	0
2	8	1	4	7	6	0	0	0	0
3	2	4	8	9	7	0	0	0	0
5	7	6	3	4	1	0	0	0	0
5	4	3	9	1	0	0	0	0	0
6	8	1	7	2	4	0	0	0	0
7	6	4	3	1	5	2	0	0	0
7	2	9	5	6	4	0	0	0	0
1	8	3	9	7	2	0	0	0	0

PSO2.in <Input File for processing time>

Table 4.4

95.501	6.163	72.114	53.212	74.316	63.239	57.031	50.381	0.000	0.000
87.329	45.583	64.886	83.452	95.812	21.379	82.908	0.000	0.000	0.000
24.627	88.957	48.776	81.253	56.357	55.305	0.000	0.000	0.000	0.000
57.929	6.076	55.565	11.460	85.340	37.945	0.000	0.000	0.000	0.000
32.497	31.292	28.370	50.532	95.679	73.998	0.000	0.000	0.000	0.000
40.238	87.458	99.855	9.460	44.779	0.000	0.000	0.000	0.000	0.000
34.534	76.959	47.903	79.053	65.366	16.913	0.000	0.000	0.000	0.000
57.840	81.564	53.269	44.371	5.719	45.818	55.350	0.000	0.000	0.000
46.746	79.875	99.578	18.826	88.727	58.780	0.000	0.000	0.000	0.000
94.133	96.447	19.503	57.031	46.066	99.713	0.000	0.000	0.000	0.000

PSO2.out<Output File>

For iter 1 gbest=914.652

For iter 2 gbest=914.652  
For iter 3 gbest=914.652  
For iter 4 gbest=914.652  
For iter 5 gbest=914.652  
For iter 6 gbest=914.652  
For iter 7 gbest=911.568  
For iter 8 gbest=911.568  
For iter 9 gbest=911.568  
For iter 10 gbest=911.568  
For iter 11 gbest=911.568  
For iter 12 gbest=911.568  
For iter 13 gbest=911.568  
For iter 14 gbest=911.568  
For iter 15 gbest=911.568  
For iter 16 gbest=911.568  
For iter 17 gbest=911.568  
For iter 18 gbest=911.568  
For iter 19 gbest=911.568  
For iter 20 gbest=911.568  
For iter 21 gbest=911.568  
For iter 22 gbest=911.568  
For iter 23 gbest=911.568  
For iter 24 gbest=911.568  
For iter 25 gbest=908.772  
For iter 26 gbest=908.772  
For iter 27 gbest=908.772  
For iter 28 gbest=908.772  
For iter 29 gbest=908.772  
For iter 30 gbest=908.772

# Chapter 5

**CONCLUSION**

## **CONCLUSION:**

Particle swarm optimization is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions. We view it as a mid-level form of A-life or biologically derived algorithm, occupying the space in nature between evolutionary search, which requires eons, and neural processing, which occurs on the order of milliseconds. Social optimization occurs in the time frame of ordinary experience - in fact, it is ordinary experience. In addition to its ties with A-life, particle swarm optimization has obvious ties with evolutionary computation. Conceptually, it seems to lie somewhere between genetic algorithms and evolutionary programming. It is highly dependent on stochastic processes, like evolutionary programming. The adjustment toward pbest and gbest by the particle swarm optimizer is conceptually similar to the crossover operation utilized by genetic algorithms. It uses the concept of fitness, as do all evolutionary computation paradigms. Unique to the concept of particle swarm optimization is flying potential solutions through hyperspace, accelerating toward "better" solutions. Other evolutionary computation schemes operate directly on potential solutions which are represented as locations in hyperspace. Much of the success of particle swarms seems to lie in the agents' tendency to hurtle past their target. Holland's chapter on the "optimum allocation of trials" reveals the delicate balance between conservative testing of known regions versus risky exploration of the unknown. It appears that the current version of the paradigm allocates trials nearly optimally. The stochastic factors allow thorough search of spaces between regions that have been found to be relatively good, and the momentum effect caused by modifying the extant velocities rather than replacing them results in overshooting, or exploration of unknown regions of the problem domain. The authors of this paper are a social psychologist and an electrical engineer. The particle swarm optimizer serves both of these fields equally well. Why is social behavior so ubiquitous in the animal kingdom? Because it optimizes. What is a good way to solve engineering optimization problems? Modeling social behavior. Much further research remains to be conducted on this simple new concept and paradigm. The goals in developing it have been to keep it simple and robust, and we seem to have succeeded at that. The algorithm is written in a very few lines of code, and requires only specification of the problem and a few parameters in order to solve it. This algorithm

belongs ideologically to that philosophical school that allows wisdom to emerge rather than trying to impose it, that emulates nature rather than trying to control it, and that seeks to make things simpler rather than more complex. Once again nature has provided us with a technique for processing information that is at once elegant and versatile.



## **REFERENCES:**

- ❖ Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proc. IEEE int'l conf. on neural networks Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.
- ❖ Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the sixth international symposium on micro machine and human science pp. 39-43. IEEE service center, Piscataway, NJ, Nagoya, Japan, 1995.
- ❖ Eberhart, R. C. and Shi, Y. Particle swarm optimization: developments, applications and resources. Proc. congress on evolutionary computation 2001 IEEE service center, Piscataway, NJ., Seoul, Korea., 2001.
- ❖ Eberhart, R. C. and Shi, Y. Evolving artificial neural networks. Proc. 1998 Int'l Conf. on neural networks and brain pp. PL5-PL13. Beijing, P. R. China, 1998.
- ❖ Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization. Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf., Springer-Verlag, Berlin, San Diego, CA., 1998.
- ❖ Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization. Evolutionary Programming VII: Proc. EP 98 pp. 591-600. Springer-Verlag, New York, 1998.
- ❖ Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation pp. 69-73. IEEE Press, Piscataway, NJ, 1998